TCP Connection Latency: TCP is a connection based, reliable, byte stream oriented protocol. As part of this reliability, a connection must be established before any data may transfer. The connection must be established by a “three way had shake,” an exchange of packets when the client attempts to connect to the server. The benchmark times the creation and connection of a TCP/IP socket to a remote host. Care is taken in that time does not include any other overhead.

6.6.2.6 File System Latency

File system latency is defined as the time it takes to create or delete a zero length file. The benchmark creates 1000, zero to 10K byte sized files, and then deletes them. DAISy and HEAT results are in Table 16.

System	file system, size = 0K (lat_fs)	
	creation	removal
DAISY systems		
FreeBSD/i586 (p5-90)	50	120
HEAT systems		
DEC Alpha	44	88
HP 9000/735	24	30
IBM RS6000	59	61
SUN SS10	29	70
SGI IRIX	58	67

Table 16. lat_fs results (microseconds).

6.6.2.7 Page Fault Latency

System	page fault (lat_pagefault)
DAISY systems	
FreeBSD/i586 (p5-90)	26
HEAT systems	
DEC Alpha	16592
HP 9000/735	22020
IBM RS6000	7464
SUN SS10	107979
SGI IRIX	21361

Table 17. lat_pagefault results (microseconds).

Page fault latency is measures how fast the file system can pagefault in a page that is not in memory. For the HEAT systems the file system is NFS mounted, therefore this test measures the cost of going across NFS.

6.6.2.8 Memory Mapping Latency

Memory mapping is an alternative to traditional read and write interfaces. It is the process to making a file part of a processes address space, allowing direct access to the file’s pages. Memory mapping is extensively used for linking in shared libraries at run time. The lat_mmap benchmark measures the speed, at which mappings can be created as well as removed.

System	memory mapping (lat_mmap) 4MB
DAISY systems	
FreeBSD/i586 (p5-90)	160
HEAT systems	
DEC Alpha	117
HP 9000/735	100
IBM RS6000	270
SUN SS10	148
SGI IRIX	600

Table 18. lat_mmap results (microseconds).

6.7 Bonnie: Disk performance benchmark

The Bonnie [24] disk performance benchmark measures the local disk performance. Table 19 shows the DAISy and **HEAT** disk performance benchmarks for sequential output on a per character and a block size measure. The benchmark size was 100MB.

		Sequential Output						Sequential Input				Random	
		Per Char		Block		Rewrite		Per Char		Block		Seeks	
Machine	Mb	K/sec	%cpu	K/sec	%cpu	K/sec	%cpu	K/sec	%cpu	K/sec	%cpu	/sec	%cpu
P5-90	100	2059	93.6	2429	57.6	971	18.7	1302	49.6	2280	29.9	156.5	13.9
DEC Alpha	100	3425	95.6	3491	14.5	1538	6.8	3497	96.9	3574	8.6	101.6	3.6
HP 735	100	1568	60.6	1492	31.7	610	4.5	1492	56.9	1530	7.5	141	6
IBM RS6K	100	1459	96.5	1558	13.3	530	6.4	1123	89.2	1939	13.2	44.8	6
SGI IRIX	100	1767	95.4	3307	26.5	1320	13.5	1336	81.6	2806	15.8	62.9	5
SUN SS10	100	1503	71.7	1552	15.4	523	8.4	1415	81.9	2275	20.5	71.9	6.6

Table 19. Bonnie results (100MB benchmark size).

6.8 Netperf suite: A network performance benchmark

Netperf [25] is a suite of benchmarks used to measure various aspects of networking performance and designed around the client/server model. The primary focus is on bulk data transfer and request/response performance using either TCP or UDP and the Berkeley Sockets interface. All benchmarks are run for an elapsed time of ~60 seconds. The various Netperf performance benchmarks fall into two categories: (a) stream, and (b) request/response.

The most common use of the Netperf suite is measuring bulk data transfer performance. This is referred to as “stream” or “unidirectional stream” performance. Basically these tests measure how fast one system can send data to another and/or how fast that other system can receive it. The “stream” scripts that were run to produce the results include: tcp_range_script, tcp_stream_script, and udp_stream_script.

Netperf request/response performance is quoted as “transactions/sec” for a given request and response size. A transaction is defined as the exchange of a single request and a single response. From a transaction rate, one can infer one way and round trip average latency. The "request/response" scripts that were run include: tcp_rr_script and udp_rr_script.

As mentioned above, Netperf is based around the client/server model. Therefore, there are two executables: (a) Netperf, and (b) netserver. The netserver program will be invoked at the server node, and the Netperf program will be invoked at the client node. When Netperf is executed, the first thing that will happen is an establishment of control to the remote host. This connection is used to pass test configuration information and results to and from the remote system. Regardless of the type of test being run, the control connection will be a TCP sockets connection. Next a separate connection is opened using the APIs and appropriate protocols for the test, and the test will be run. While the test is being run there is no traffic on the original control connection..

The results below from the DAISy and HEAT clusters will be condensed, but are in full in the appendices. It should be noted that results are from the original scripts that come from the Netperf http site.

6.8.1 Stream

6.8.1.1 TCP Range

tcp_range_script is an implementation of the stream benchmark over TCP. The local send size ranges from 1 to 65536 bytes with the local/remote send and receive socket buffer sizes of each set to 32768 bytes. With to send and receive socket buffer sizes remaining constant, the output shows the throughput (Mb/s) as a function of range (bytes). For the results in Table 20: recv & send socket size = 32768 bytes, send message size = 65536 bytes.

6.8.1.2 TCP Stream

tcp_stream_script is an implementation of the stream benchmark over TCP The local send size ranges from 4096 to 32768 bytes with the local/remote send and receive socket buffer sizes of each ranging from 8102 to 57344 bytes. With the send and receive socket buffer sizes not remaining constant, the output shows throughput (Mb/s)t as a function of range (bytes). For the results in Table 20: recv & send socket size = 57344, send message size = 32768.

6.8.1.3 UDP Stream

udp_stream_script is an implementation of the stream benchmark over UDP. The difference between udp_stream and tcp_stream is that the send size cannot be larger than the smaller of the local and remote socket buffer sizes. The local send size ranges from 64 to 1472 bytes with the local/remote send ad receive socket buffer sizes of each remain constant at 32768 bytes. With the send and receive socket buffer sizes remaining constant, the output shows throughput (Mb/s), as a function of range (bytes) for both send and receive. For the results in Table 20: socket size = 32768, message size = 1472.

System	Network	TCP range	TCP stream	UDP stream send	receive
DAISY systems					
FreeBSD/i586 (p5-90)	10baseT	5.38	5.19	5.73	5.73
FreeBSD/i586 (p5-90)	100baseT	50.06	50.64	68.5	35.28
HEAT systems					
DEC Alpha	fddi	79.95	85.87	89.23	37.14
HP 9000/735	fddi	71.53	79.31	87.38	52.18
SGI IRIX	fddi	60.53	71.08	69.66	11.45

Table 20. tcp_range_script, tcp_stream_script, udp_stream_script results (Mb/s).

6.8.2 Request/Response

6.8.2.1 TCP Request/Response

tcp_rr_script is an implementation of the request/response benchmark over TCP. The request/response sizes are varied with the local/remote send and receive socket buffer sizes of each being the default of that particular system. With the local/remote send and receive socket buffer sizes remaining constant (the default), the output shows performance (transactions/s) as a function of request/response sizes (bytes). For the results in Table 21: send & recv socket = default bytes, request/resp. size = 1/1.

6.8.2.2 UDP Request/Response

udp_rr_script is an implementation of the request/response benchmark over UDP. The request/response sizes are varied with the local/remote send and receive socket buffer sizes of each being the default of that particular system. With the local/remote send and receive socket buffer sizes remaining constant (the default), the output shows performance (transactions/s) as a functions of request/response sizes (bytes).). For the results in Table 21: send & recv socket = default bytes, request/resp. size = 1/1.

System	Network	TCP request/response	UDP request/response
DAISY systems			
FreeBSD/i586 (p5-90)	10baseT	1331	1659
FreeBSD/i586 (p5-90)	100baseT	1638	2096
HEAT systems			
DEC Alpha	fddi	1772	1937
HP 9000/735	fddi	2423	2473
SGI IRIX	fddi	993	34

Table 21. tcp_rr_script, udp_rr_script results (transactions/s).

6.9 The PVM Timing Example

The PVM timing example [26] is a simple program used to illustrate how to measure network bandwidth and latency under PVM. It is a part of the example programs that are included in the PVM distribution. Not only is it a good I/O performance benchmark, but is a good benchmark for testing out PVM.

System	Network	PVM timing ex.
DAISY systems		
FreeBSD/i586 (p5-90)	10baseT	0.695
FreeBSD/i586 (p5-90)	100baseT	5.284
HEAT systems		
DEC Alpha	fddi	8.242
HP 9000/735	fddi	9.5
SGI IRIX	fddi	6.66

Table 22. PVM timing example results (avg. bytes/microseconds).

6.10 NFS Performance

Measuring NFS performance requires measuring both client and server performance. Table 23 shows NFS server performance using the Bonnie benchmark [24]. The Bonnie benchmark was run on a client node to the user home directory file system on the master node d-00, using the DAISy 10BASE-2 operational network.

The NFS client performance was measured by copying a large file from the master node to /dev/null on the client system. For an 11MB file the best rate measured at 602KB/s.

		Sequential Output						Sequential Input				Random	
		Per Char		Block		Rewrite		Per Char		Block		Seeks	
Machine	Mb	K/sec	%cpu	K/sec	%cpu	K/sec	%cpu	K/sec	%cpu	K/sec	%cpu	/sec	%cpu
P5-90	100	131	4.5	128	1.1	12	0.3	660	24.4	571	4.7	25.4	3.8
DEC Alpha	100	519	13.6	756	3.4	350	1.7	1065	27	1201	2.6	99.2	5.2
HP 735	100	645	25.4	679	11.9	330	2.3	835	31.6	1032	4.9	106.6	4.8
IBM RS6K	100	562	38.9	692	8.2	289	5	405	33.5	1440	11.6	35.4	2.9
SGI IRIX	100	258	16.6	287	3.4	188	6	174	14.4	425	7.8	26.6	5.8
SUN SS10	100	407	20.5	581	8.1	252	6	463	27.5	615	6.3	47	8.3

Table 23. Bonnie NFS Server Performance results (100MB benchmark size).

6.11 NAS Parallel Benchmarks 1.0, PVM versions

The NAS Parallel Benchmarks 1.0 (NPB 1.0) [10], [27] consisted of eight benchmark problems. Five of these were kernel benchmarks and three were simulated computational fluid dynamics (CFD) applications. We obtained the PVM versions of the NPB1.0 from the HENSA UNIX Parallel Archive at the Internet URL address *http://www.hensa.ac.uk/parallel/environments/pvm3/NAS-benchmarks/*. Unfortunately, only one of the eight benchmarks were able to run in class A mode on the DAISy cluster. This was the "embarrassingly parallel" benchmark EP. The remaining NPB 1.0 PVM benchmarks were able to compile and run successfully on the DAISy and HEAT machines in sample mode. These will not be discussed here, but the appendix contains these results. As a note, the recently published NPB 2.0 [7] benchmarks running under MPI have run semi-successfully on DAISy and the results are within this paper.

6.11.1 Kernel EP

Briefly, Kernel EP executes 2^28 iterations of a loop in which a pair of random numbers are generated and tested for whether Gaussian random deviates can be made from them according to a specific scheme. The number of pairs of the Gaussian's in 10 successive square annuli are tabulated. The psudorandom number generator used in this , and in all NAS benchmarks which call for random numbers, is of the linear congruential recursion type. This kernel is viewed and named as an "embarrassingly parallel" application. In other words, improved throughput rather than turn around time. Based on the partitionability of the problem, no data or functional dependencies are incurred, and there is little or no communication between processors.

Figure 18 shows the scalability of the EP Kernel benchmark on the DAISy cluster. Note that the time it takes to execute the benchmark on one processor is almost exactly fifteen time slower than it would be to execute the benchmark on 15 processors, hence, "embarrassingly parallel". Table 24 shows the results from the DAISy and HEAT clusters using 8 nodes each.

size = 2^28 # of processors	Benchmark time (sec) p5-90, 100Mb/s sw
15	287.97
14	307.83
13	331.45
12	358.17
11	391.82
10	430.70
9	481.27
8	540.89
7	616.46
6	718.62
5	865.28
4	1076.66
3	1440.13
2	2152.87
1	4304.07

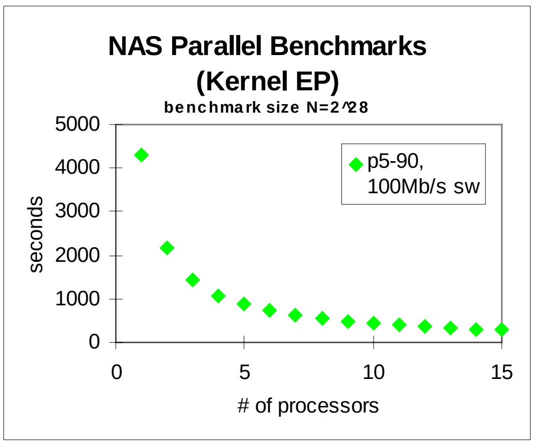

Figure 18. Kernel EP results on 1 to 15 DAISy nodes.

System	Network	Kernel EP
DAISY systems		
FreeBSD/i586 (p5-90)	10baseT	537
FreeBSD/i586 (p5-90)	100baseT	541
HEAT systems		
DEC Alpha	fddi	408
IBM RS6K	fddi	775
SGI IRIX	fddi	1193